CLAIMS

We claim:

1. In a computer system for automated software testing to determine whether or not software operates as intended, a method of applying one or more tests to one or more software components without having to specify each test or test environment explicitly, the method comprising acts of:

reading a test file containing (i) one or more code sections that identify one or more software components to test, (ii) one or more location sections that identify one or more locations to run the one or more software components, (iii) an expandable variation section that associates the one or more software components with the one or more locations, and (iv) an expansion section that defines rules for expanding the expandable variation section;

expanding the expandable variation section to create one or more expanded variation sections, each associating a particular one of the one or more software components with a particular one of the one or more locations; and

for each particular software component and location, calling a code setup routine to prepare for executing the particular software component, a code run routine to execute the particular software component for testing, and a code cleanup routine to reverse changes made by the code setup and run routines as needed.

2. A method as recited in claim 1, wherein the test file contains one or more group sections for grouping each of the one or more expanded variation sections, the method further comprising an act of, for each of the one or more group sections, calling a group setup routine prior to executing any expanded variation section within the corresponding group section, and calling a group cleanup routine after executing any expanded variation section within the corresponding group section.

3. A method as recited in claim 2, wherein the test file contains one or more set sections for logically grouping the one or more expanded variation sections.

4. A method as recited in claim 1, further comprising an act of calling an optional setup routine for a corresponding group section.

5. A method as recited in claim 1, further comprising an act of passing context information into a code section.

6. A method as recited in claim 1, wherein the expansion section identifies a list expansion algorithm which generates permutations from a list of one or more placeholders and list of one or more substitution strings.

7. A method as recited in claim 1, wherein the test file conforms to an eXtensible Markup Language (XML) schema.

8. In a computer system for automated software testing to determine whether or not software operates as intended, a computer program product comprising one or more computer readable media carrying computer executable instructions that implement a method of applying one or more tests to one or more software components without having to specify each test or test environment explicitly, the method comprising acts of:

reading a test file containing (i) one or more code sections that identify one or more software components to test, (ii) one or more location sections that identify one or more locations to run the one or more software components, (iii) an expandable variation section that associates the one or more software components with the one or more locations, and (iv) an expansion section that defines rules for expanding the expandable variation section;

expanding the expandable variation section to create one or more expanded variation sections, each associating a particular one of the one or more software components with a particular one of the one or more locations; and

for each particular software component and location, calling a code setup routine to prepare for executing the particular software component, a code run routine to execute the particular software component for testing, and a code cleanup routine to reverse changes made by the code setup and run routines as needed.

9. A computer program product as recited in claim 8, the method further comprising an act of calling an optional setup routine for a corresponding code section.

10. A computer program product as recited in claim 8, the method further comprising an act of passing synchronization information into a group section.

11. A computer program product as recited in claim 8, the method further comprising an act of writing testing information to a log.

12. A computer program product as recited in claim 8, wherein the expansion section identifies a pair-wise expansion algorithm that generates unique combinations from a list of substitution strings and a number of substitution strings to be included in each unique combination.

13. A computer program product as recited in claim 8, wherein the test file contains one or more parameter sections, the method further comprising an act of accessing data within at least one of the parameter sections from a group section within a scope that includes the at least one parameter section and the group section.

14. A computer program product as recited in claim 8, the method further comprising an act of writing one or more actual test results for use in determining either a pass or fail result for a code section.

15. A computer program product as recited in claim 14, wherein the test file contains one or more expected test results, and wherein the test file references one or more result comparators, the method further comprising an act of comparing the one or more actual test results with the one or more expected test results to determine the pass or fail result for the code section.

16. In a computer system for automated software testing to determine whether or not software operates as intended, a method of applying one or more tests to one or more software components without having to specify each test or test environment explicitly, the method comprising step for:

loading and parsing a test file containing (i) one or more code sections that identify one or more software components to test, (ii) one or more location sections that identify one or more locations to run the one or more software components, (iii) an expandable variation section that associates the one or more software components with the one or more locations, and (iv) an expansion section that defines rules for expanding the expandable variation section;

creating one or more expanded variation sections from the expandable variation section, each associating a particular one of the one or more software components with a particular one of the one or more locations; and

for each particular software component and location, executing setup instructions to prepare for running the particular software component, the particular software component being tested, and cleanup instructions following execution of the software component being tested.

17. A method as recited in claim 16, further comprising an act of calling an optional cleanup routine for a corresponding group section.

18. A method as recited in claim 16, further comprising a step for sharing context information between a group and a portion.

19. A method as recited in claim 16, wherein the expansion section identifies a data file expansion algorithm that replaces one or more placeholders with a comma separated list of substitutes from a data file.

20. A method as recited in claim 16, wherein the test file contains one or more parameter sections, the method further comprising an act of accessing data within at least one of the parameter sections from a code section within a scope that includes the at least one parameter section and the code section.

21. A method as recited in claim 16, further comprising a step for recording one or more actual test results for use in determining either a pass or fail result for a code section.

22. A method as recited in claim 21, wherein the test file contains one or more expected test results, and wherein the test file references one or more result comparators, the method further comprising a step for using the one or more result comparators to compare the one or more actual test results with the one or more expected test results to determine the pass or fail result for the code section.

23. In a computer system for automated software testing to determine whether or not software operates as intended, a computer program product comprising one or more computer readable media carrying computer executable instructions that implement a method of applying one or more tests to one or more software components without having to specify each test or test environment explicitly, the method comprising steps for:

loading and parsing a test file containing (i) one or more code sections that identify one or more software components to test, (ii) one or more location sections that identify one or more locations to run the one or more software components, (iii) an expandable variation section that associates the one or more software components with the one or more locations, and (iv) an expansion section that defines rules for expanding the expandable variation section;

creating one or more expanded variation sections from the expandable variation section, each associating a particular one of the one or more software components with a particular one of the one or more locations; and

for each particular software component and location, executing setup instructions to prepare for running the particular software component, the particular software component being tested, and cleanup instructions following execution of the software component being tested.

24. A computer program product as recited in claim 23, the method further comprising an act of calling an optional cleanup routine for a corresponding code section.

25. A computer program product as recited in claim 23, wherein the test file contains at least two code sections, the method further comprising a step for sharing synchronization information between the at least two code sections.

26. A computer program product as recited in claim 23, the method further comprising a step for recording testing information in a log.

27. A computer program product as recited in claim 26, wherein testing information is recorded in the log based on the selection of one or more logging levels comprising at least one of an always level, an error level, a warn level, and a trace level.

28. A computer program product as recited in claim 23, wherein the expansion section identifies a range expansion algorithm that substitutes all values within a range defined by a lower and upper limit for a placeholder.

29. For an automated software testing system used to determine whether or not software operates as intended, a computer program product comprising one or more computer readable media for applying one or more tests to one or more software components without explicitly having to specify each test or test environment, the computer program product carrying a test file that comprises:

one or more code sections that identify one or more software components to test;

one or more location sections that identify one or more locations to run the one or more software components;

at least one expandable variation section that associates the one or more software components with the one or more locations; and

an expansion section that defines rules for creating one or more expanded variation sections and associating a particular software component with a particular location.

30. A computer program product as recited in claim 29, wherein the test file further comprises one or more group sections for grouping each of the one or more expanded variation sections.

31. A computer program product as recited in claim 30, wherein the test file contains one or more parameter sections for access from a code or group section within a scope that includes the at least one parameter section and the code or group section.

32. A computer program product as recited in claim 29, wherein the test file further comprises one or more set sections for logically grouping the one or more expanded variation sections.

33. A computer program product as recited in claim 29, wherein the expansion section identifies an expansion algorithm comprising at least one of (i) a list expansion algorithm which generates permutations from a list of one or more placeholders and list of one or more substitution strings, (ii) a pair-wise expansion algorithm that generates unique combinations from a list of substitution strings and a number of substitution strings to be included in each unique combination, (iii) a data file expansion algorithm that replaces one or more placeholders with a comma separated list of substitutes from a data file, and (iv) a range expansion algorithm that substitutes all values within a range defined by a lower and upper limit for a placeholder.

34. A method as recited in claim 21, wherein the test file contains one or more expected test results, and wherein the test file references one or more result comparators for comparing one or more actual test results with the one or more expected test results to determine a pass or fail result for a code section.

35. A computer program product as recited in claim 29, wherein the test file conforms to an eXtensible Markup Language (XML) schema.